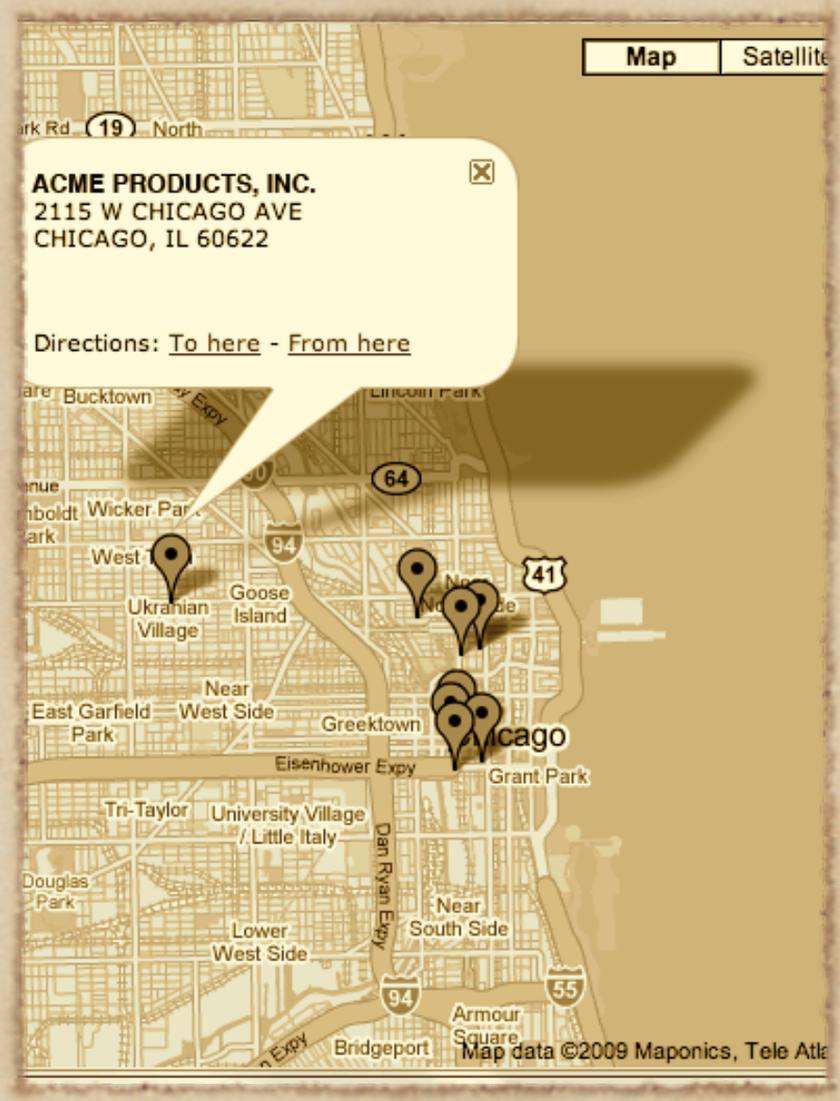
FOX-GEO

Leveraging the
Google Maps API
from FoxPro



Session Outline

Session Outline

- GeoCoding Our Data

Session Outline

- GeoCoding Our Data
- Javascript generation

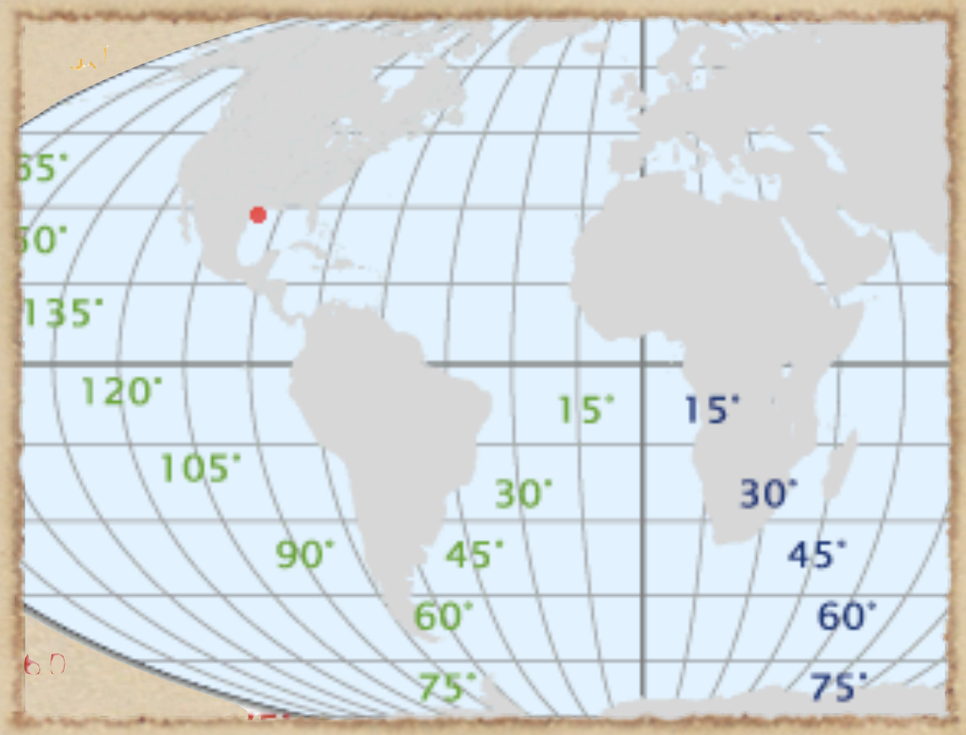
Session Outline

- GeoCoding Our Data
- Javascript generation
- Interact with Google Maps API

Session Outline

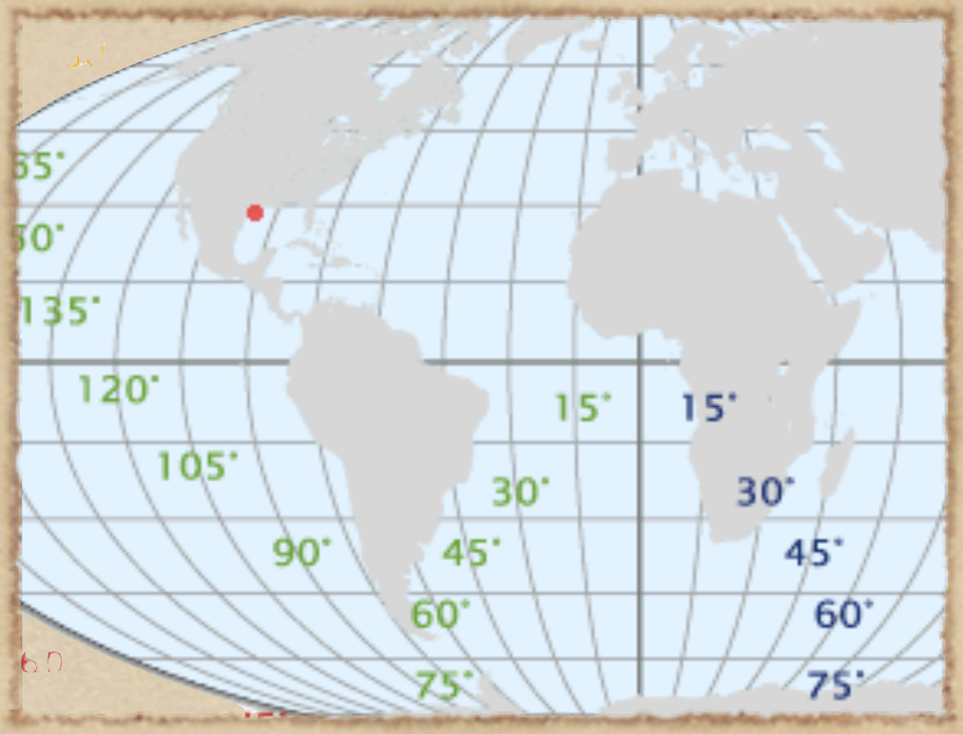
- GeoCoding Our Data
- Javascript generation
- Interact with Google Maps API
- Code Walkthrough

What is GeoCoding?



What is GeoCoding?

- the process of finding geographic coordinates (lattitude and longitude) from other geographic data such as addresses or zip codes.

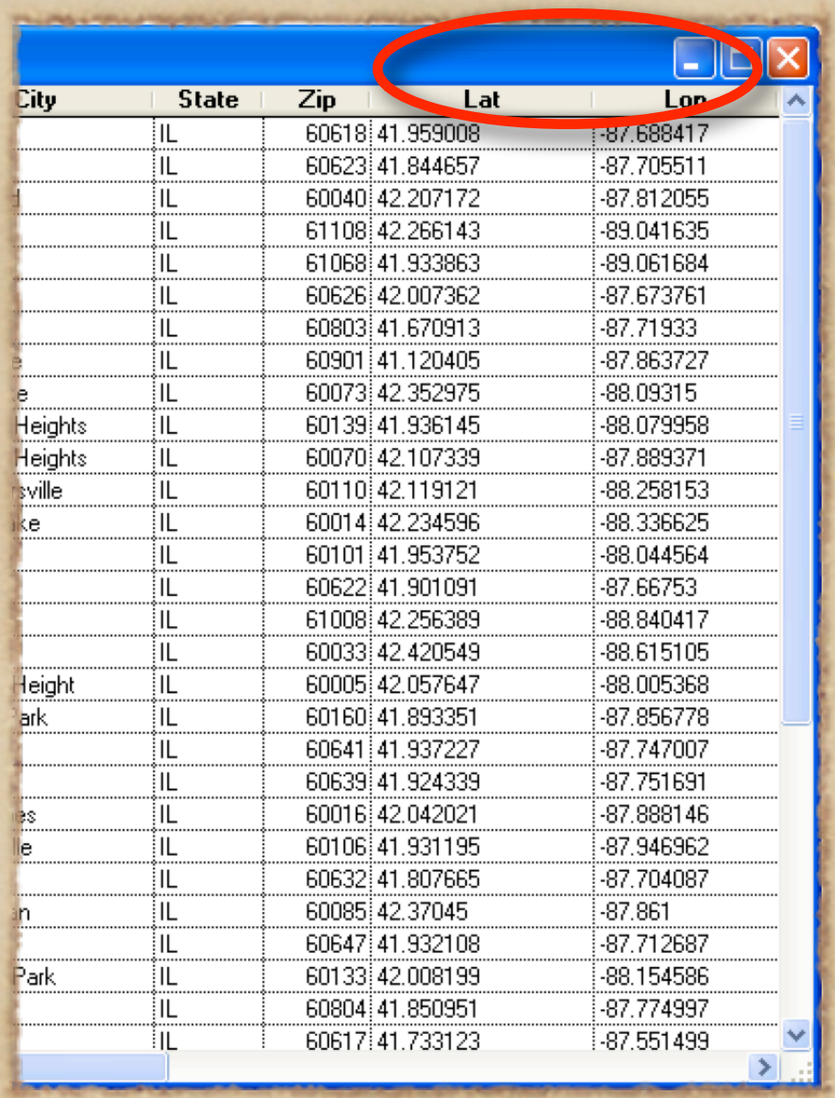


GeoCoding our data

City	State	Zip	Lat	Lon
	IL	60618	41.959008	-87.688417
	IL	60623	41.844657	-87.705511
	IL	60040	42.207172	-87.812055
	IL	61108	42.266143	-89.041635
	IL	61068	41.933863	-89.061684
	IL	60626	42.007362	-87.673761
	IL	60803	41.670913	-87.71933
	IL	60901	41.120405	-87.863727
e	IL	60073	42.352975	-88.09315
Heights	IL	60139	41.936145	-88.079958
Heights	IL	60070	42.107339	-87.889371
sville	IL	60110	42.119121	-88.258153
ke	IL	60014	42.234596	-88.336625
	IL	60101	41.953752	-88.044564
	IL	60622	41.901091	-87.66753
	IL	61008	42.256389	-88.840417
	IL	60033	42.420549	-88.615105
Height	IL	60005	42.057647	-88.005368
ark	IL	60160	41.893351	-87.856778
	IL	60641	41.937227	-87.747007
	IL	60639	41.924339	-87.751691
es	IL	60016	42.042021	-87.888146
e	IL	60106	41.931195	-87.946962
	IL	60632	41.807665	-87.704087
n	IL	60085	42.37045	-87.861
	IL	60647	41.932108	-87.712687
Park	IL	60133	42.008199	-88.154586
	IL	60804	41.850951	-87.774997
	IL	60617	41.733123	-87.551499

GeoCoding our data

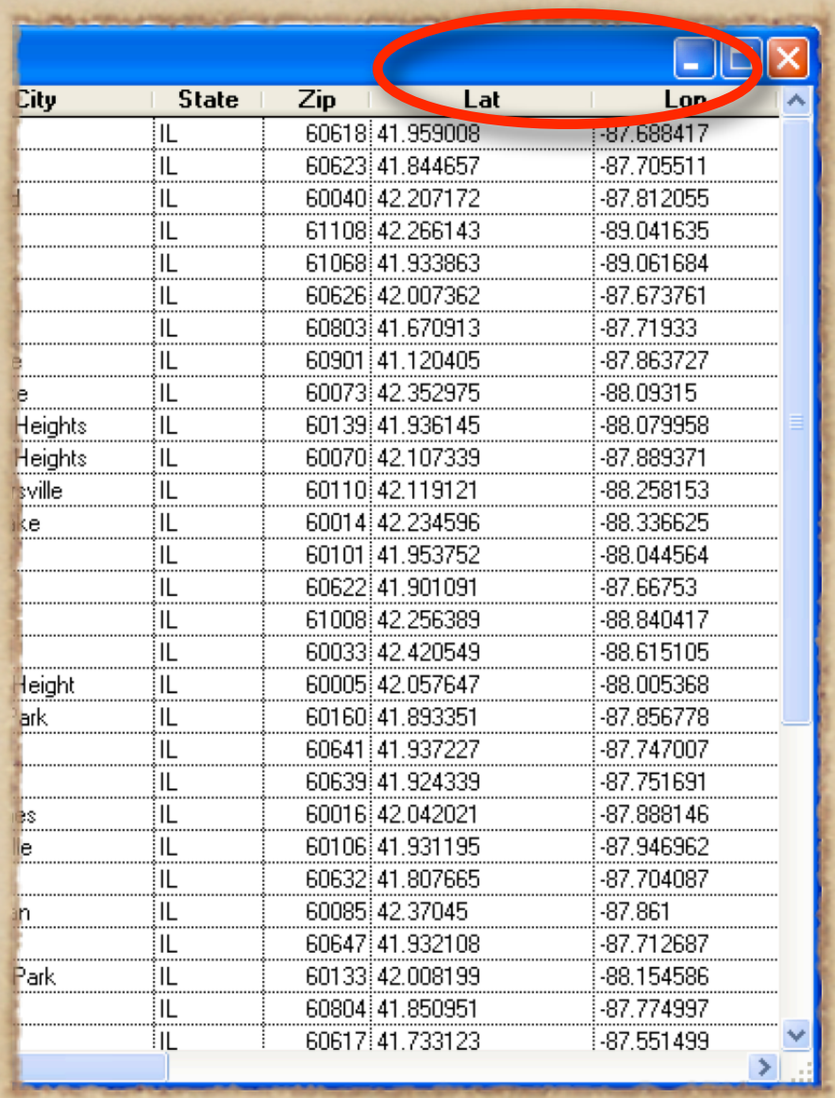
- Add two fields, named lat, lon c(10)



City	State	Zip	Lat	Lon
	IL	60618	41.959008	-87.688417
	IL	60623	41.844657	-87.705511
	IL	60040	42.207172	-87.812055
	IL	61108	42.266143	-89.041635
	IL	61068	41.933863	-89.061684
	IL	60626	42.007362	-87.673761
	IL	60803	41.670913	-87.71933
	IL	60901	41.120405	-87.863727
e	IL	60073	42.352975	-88.09315
Heights	IL	60139	41.936145	-88.079958
Heights	IL	60070	42.107339	-87.889371
sville	IL	60110	42.119121	-88.258153
ke	IL	60014	42.234596	-88.336625
	IL	60101	41.953752	-88.044564
	IL	60622	41.901091	-87.66753
	IL	61008	42.256389	-88.840417
	IL	60033	42.420549	-88.615105
Height	IL	60005	42.057647	-88.005368
ark	IL	60160	41.893351	-87.856778
	IL	60641	41.937227	-87.747007
	IL	60639	41.924339	-87.751691
es	IL	60016	42.042021	-87.888146
e	IL	60106	41.931195	-87.946962
	IL	60632	41.807665	-87.704087
n	IL	60085	42.37045	-87.861
	IL	60647	41.932108	-87.712687
Park	IL	60133	42.008199	-88.154586
	IL	60804	41.850951	-87.774997
	IL	60617	41.733123	-87.551499

GeoCoding our data

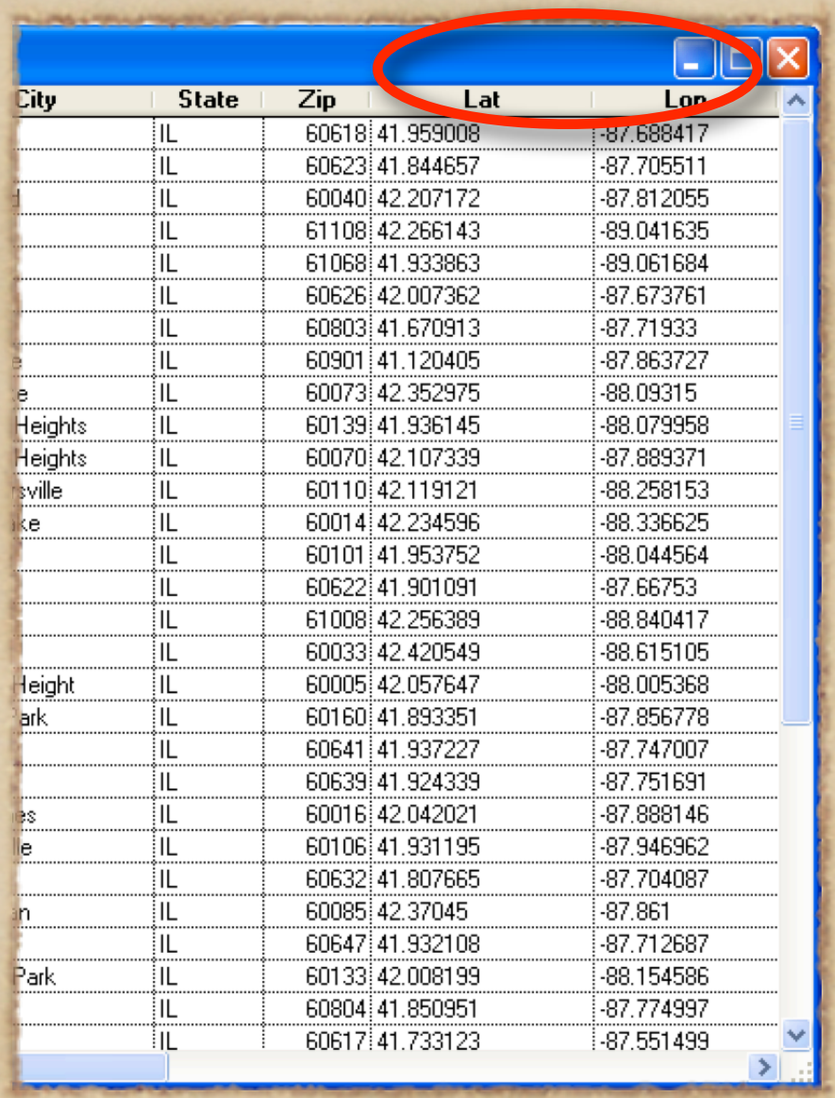
- Add two fields, named lat, lon c(10)
- Modify BatchGeo's table and field names.



City	State	Zip	Lat	Lon
	IL	60618	41.959008	-87.688417
	IL	60623	41.844657	-87.705511
	IL	60040	42.207172	-87.812055
	IL	61108	42.266143	-89.041635
	IL	61068	41.933863	-89.061684
	IL	60626	42.007362	-87.673761
	IL	60803	41.670913	-87.71933
	IL	60901	41.120405	-87.863727
e	IL	60073	42.352975	-88.09315
Heights	IL	60139	41.936145	-88.079958
Heights	IL	60070	42.107339	-87.889371
sville	IL	60110	42.119121	-88.258153
ke	IL	60014	42.234596	-88.336625
	IL	60101	41.953752	-88.044564
	IL	60622	41.901091	-87.66753
	IL	61008	42.256389	-88.840417
	IL	60033	42.420549	-88.615105
Height	IL	60005	42.057647	-88.005368
ark	IL	60160	41.893351	-87.856778
	IL	60641	41.937227	-87.747007
	IL	60639	41.924339	-87.751691
es	IL	60016	42.042021	-87.888146
e	IL	60106	41.931195	-87.946962
	IL	60632	41.807665	-87.704087
n	IL	60085	42.37045	-87.861
	IL	60647	41.932108	-87.712687
Park	IL	60133	42.008199	-88.154586
	IL	60804	41.850951	-87.774997
	IL	60617	41.733123	-87.551499

GeoCoding our data

- Add two fields, named lat, lon c(10)
- Modify BatchGeo's table and field names.
- Run it.



City	State	Zip	Lat	Lon
	IL	60618	41.959008	-87.688417
	IL	60623	41.844657	-87.705511
	IL	60040	42.207172	-87.812055
	IL	61108	42.266143	-89.041635
	IL	61068	41.933863	-89.061684
	IL	60626	42.007362	-87.673761
	IL	60803	41.670913	-87.71933
	IL	60901	41.120405	-87.863727
e	IL	60073	42.352975	-88.09315
Heights	IL	60139	41.936145	-88.079958
Heights	IL	60070	42.107339	-87.889371
sville	IL	60110	42.119121	-88.258153
ke	IL	60014	42.234596	-88.336625
	IL	60101	41.953752	-88.044564
	IL	60622	41.901091	-87.66753
	IL	61008	42.256389	-88.840417
	IL	60033	42.420549	-88.615105
Height	IL	60005	42.057647	-88.005368
ark	IL	60160	41.893351	-87.856778
	IL	60641	41.937227	-87.747007
	IL	60639	41.924339	-87.751691
es	IL	60016	42.042021	-87.888146
e	IL	60106	41.931195	-87.946962
	IL	60632	41.807665	-87.704087
n	IL	60085	42.37045	-87.861
	IL	60647	41.932108	-87.712687
Park	IL	60133	42.008199	-88.154586
	IL	60804	41.850951	-87.774997
	IL	60617	41.733123	-87.551499

```
DECLARE Sleep IN kernel32 INTEGER dwMillisecond

gm_apikey = ""
USE offices
LOCAL loHttp AS MSXML2.ServerXMLHTTP.3.0
loHTTP = CreateObject("MSXML2.ServerXMLHTTP.3.0")
lcGMURL = "http://maps.google.com/maps/geo?q=<<lcAddress>>&output=csv&oe=utf8&sensor=

SCAN FOR VAL(lat) = 0
    *-- do some address scrubbing here
    *-- PO Boxes --> zipcode only
    IF "BOX" $ UPPER(address_2)
        lcAddress = TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zip)
    ELSE
        lcAddress = TRIM(address_2)+", "+TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zi
    ENDIF
    lcAddress = urlencode(lcAddress)
    loHttp.open("GET",TEXTMERGE(lcGMUrl),.f.)
    loHttp.send()
    lcResult = loHttp.responseText

    IF VAL(lcResult) == 200
        SELECT offices
        REPLACE lat WITH GETWORDNUM(lcResult,3,",")
        REPLACE lon WITH GETWORDNUM(lcResult,4,",")
    ENDIF
    Sleep(2000) && wait for two seconds to avoid triggering google's transaction limi
ENDSCAN
```

BatchGeo.prg

```
DECLARE Sleep IN kernel32 INTEGER dwMillisecond

gm_apikey = ""
USE offices
LOCAL loHttp AS MSXML2.ServerXMLHTTP.3.0
loHTTP = CreateObject("MSXML2.ServerXMLHTTP.3.0")
lcGMURL = "http://maps.google.com/maps/geo?q=<<lcAddress>>&output=csv&oe=utf8&sensor=

SCAN FOR VAL(lat) = 0
    *-- do some address scrubbing here
    *-- PO Boxes --> zipcode only
    IF "BOX" $ UPPER(address_2)
        lcAddress = TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zip)
    ELSE
        lcAddress = TRIM(address_2)+", "+TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zi
    ENDIF
    lcAddress = urlencode(lcAddress)
    loHttp.open("GET",TEXTMERGE(lcGMUrl),.f.)
    loHttp.send()
    lcResult = loHttp.responseText

    IF VAL(lcResult) == 200
        SELECT offices
        REPLACE lat WITH GETWORDNUM(lcResult,3,",")
        REPLACE lon WITH GETWORDNUM(lcResult,4,",")
    ENDIF
    Sleep(2000) && wait for two seconds to avoid triggering google's transaction limi
ENDSCAN
```

BatchGeo.prg

```
DECLARE Sleep IN kernel32 INTEGER dwMillisecond

gm_apikey = ""
USE offices
LOCAL loHttp AS MSXML2.ServerXMLHTTP.3.0
loHTTP = CreateObject("MSXML2.ServerXMLHTTP.3.0")
lcGMURL = "http://maps.google.com/maps/geo?q=<<lcAddress>>&output=csv&oe=utf8&sensor=

SCAN FOR VAL(lat) = 0
    *-- do some address scrubbing here
    *-- PO Boxes --> zipcode only
    IF "BOX" $ UPPER(address_2)
        lcAddress = TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zip)
    ELSE
        lcAddress = TRIM(address_2)+", "+TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zi
    ENDIF
    lcAddress = urlencode(lcAddress)
    loHttp.open("GET",TEXTMERGE(lcGMUrl),.f.)
    loHttp.send()
    lcResult = loHttp.responseText

    IF VAL(lcResult) == 200
        SELECT offices
        REPLACE lat WITH GETWORDNUM(lcResult,3,",")
        REPLACE lon WITH GETWORDNUM(lcResult,4,",")
    ENDIF
    Sleep(2000) && wait for two seconds to avoid triggering google's transaction limi
ENDSCAN
```

BatchGeo.prg

```
DECLARE Sleep IN kernel32 INTEGER dwMillisecond

gm_apikey = ""
USE offices
LOCAL loHttp AS MSXML2.ServerXMLHTTP.3.0
loHTTP = CreateObject("MSXML2.ServerXMLHTTP.3.0")
lcGMURL = "http://maps.google.com/maps/geo?q=<<lcAddress>>&output=csv&oe=utf8&sensor=

SCAN FOR VAL(lat) = 0
    *-- do some address scrubbing here
    *-- PO Boxes --> zipcode only
    IF "BOX" $ UPPER(address_2)
        lcAddress = TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zip)
    ELSE
        lcAddress = TRIM(address_2)+", "+TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zi
    ENDIF
    lcAddress = urlencode(lcAddress)
    loHttp.open("GET",TEXTMERGE(lcGMUrl),.f.)
    loHttp.send()
    lcResult = loHttp.responseText

    IF VAL(lcResult) == 200
        SELECT offices
        REPLACE lat WITH GETWORDNUM(lcResult,3,",")
        REPLACE lon WITH GETWORDNUM(lcResult,4,",")
    ENDIF
    Sleep(2000) && wait for two seconds to avoid triggering google's transaction limi
ENDSCAN
```

BatchGeo.prg

```
DECLARE Sleep IN kernel32 INTEGER dwMillisecond

gm_apikey = ""
USE offices
LOCAL loHttp AS MSXML2.ServerXMLHTTP.3.0
loHTTP = CreateObject("MSXML2.ServerXMLHTTP.3.0")
lcGMURL = "http://maps.google.com/maps/geo?q=<<lcAddress>>&output=csv&oe=utf8&sensor=

SCAN FOR VAL(lat) = 0
    *-- do some address scrubbing here
    *-- PO Boxes --> zipcode only
    IF "BOX" $ UPPER(address_2)
        lcAddress = TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zip)
    ELSE
        lcAddress = TRIM(address_2)+", "+TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zi
    ENDIF
    lcAddress = urlencode(lcAddress)
    loHttp.open("GET",TEXTMERGE(lcGMUrl),.f.)
    loHttp.send()
    lcResult = loHttp.responseText

    IF VAL(lcResult) == 200
        SELECT offices
        REPLACE lat WITH GETWORDNUM(lcResult,3,",")
        REPLACE lon WITH GETWORDNUM(lcResult,4,",")
    ENDIF
    Sleep(2000) && wait for two seconds to avoid triggering google's transaction limi
ENDSCAN
```

BatchGeo.prg

```
DECLARE Sleep IN kernel32 INTEGER dwMillisecond

gm_apikey = ""
USE offices
LOCAL loHttp AS MSXML2.ServerXMLHTTP.3.0
loHTTP = CreateObject("MSXML2.ServerXMLHTTP.3.0")
lcGMURL = "http://maps.google.com/maps/geo?q=<<lcAddress>>&output=csv&oe=utf8&sensor=

SCAN FOR VAL(lat) = 0
    *-- do some address scrubbing here
    *-- PO Boxes --> zipcode only
    IF "BOX" $ UPPER(address_2)
        lcAddress = TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zip)
    ELSE
        lcAddress = TRIM(address_2)+", "+TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zi
    ENDIF
    lcAddress = urlencode(lcAddress)
    loHttp.open("GET",TEXTMERGE(lcGMUrl),.f.)
    loHttp.send()
    lcResult = loHttp.responseText

    IF VAL(lcResult) == 200
        SELECT offices
        REPLACE lat WITH GETWORDNUM(lcResult,3,",")
        REPLACE lon WITH GETWORDNUM(lcResult,4,",")
    ENDIF
    Sleep(2000) && wait for two seconds to avoid triggering google's transaction limi
ENDSCAN
```

BatchGeo.prg

```
DECLARE Sleep IN kernel32 INTEGER dwMillisecond

gm_apikey = ""
USE offices
LOCAL loHttp AS MSXML2.ServerXMLHTTP.3.0
loHTTP = CreateObject("MSXML2.ServerXMLHTTP.3.0")
lcGMURL = "http://maps.google.com/maps/geo?q=<<lcAddress>>&output=csv&oe=utf8&sensor=

SCAN FOR VAL(lat) = 0
    *-- do some address scrubbing here
    *-- PO Boxes --> zipcode only
    IF "BOX" $ UPPER(address_2)
        lcAddress = TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zip)
    ELSE
        lcAddress = TRIM(address_2)+", "+TRIM(city)+", "+TRIM(state)+" "+TRANSFORM(zi
    ENDIF
    lcAddress = urlencode(lcAddress)
    loHttp.open("GET",TEXTMERGE(lcGMUrl),.f.)
    loHttp.send()
    lcResult = loHttp.responseText

    IF VAL(lcResult) == 200
        SELECT offices
        REPLACE lat WITH GETWORDNUM(lcResult,3,",")
        REPLACE lon WITH GETWORDNUM(lcResult,4,",")
    ENDIF
    Sleep(2000) && wait for two seconds to avoid triggering google's transaction limi
ENDSCAN
```

BatchGeo.prg

Javascript generation

Javascript generation

- Because Google's Map API is written in Javascript...

Javascript generation

- Because Google's Map API is written in Javascript...
- We need to add our data to Google's map by making Javascript calls to their API.

Javascript generation

- Because Google's Map API is written in Javascript...
- We need to add our data to Google's map by making Javascript calls to their API.
- From Foxpro???

Enter map.js.vml

```
<%
SELECT offices
GO TOP
lcComma = ""
SCAN FOR !EMPTY(lat)
%>
    createMarker({
        name: '<<STR
        html: '<p><b
        lon: <<TRIM(
    city: '<<trim(ci
    });
<%
ENDSCAN
%>
```

Enter map.js.vml

- Just javascript with a Foxpro Scan loop embedded in it.

```
<%
SELECT offices
GO TOP
lcComma = ""
SCAN FOR !EMPTY(lat)
%>
    createMarker({
        name: '<<STR
        html: '<p><b
        lon: <<TRIM(
    city: '<<trim(ci
    });
<%
ENDSCAN
%>
```

Enter map.js.vml

- Just javascript with a Foxpro Scan loop embedded in it.
- RenderVml executes the embedded code

```
<%
SELECT offices
GO TOP
lcComma = ""
SCAN FOR !EMPTY(lat)
%>
    createMarker({
        name: '<<STR
        html: '<p><b
        lon: <<TRIM(
    city: '<<trim(ci
    });
<%
ENDSCAN
%>
```

Enter map.js.vml

- Just javascript with a Foxpro Scan loop embedded in it.
- RenderVml executes the embedded code
- Save the resulting file as map.js

```
<%
SELECT offices
GO TOP
lcComma = ""
SCAN FOR !EMPTY(lat)
%>
    createMarker({
        name: '<<STR
        html: '<p><b
        lon: <<TRIM(
    city: '<<trim(ci
    });
<%
ENDSCAN
%>
```

Interact with the Google Maps API

Interact with the Google Maps API

- Register on their site for an API key

Interact with the Google Maps API

- Register on their site for an API key
- Add their map library to the html page.

Interact with the Google Maps API

- Register on their site for an API key
- Add their map library to the html page.
- Add our generated map.js to the html page.

Dissecting map.js

```
// create the map
map = new GMap2(document.getElementById("map"));
...
// === create a GDirections Object ===
gdir=new GDirections(map, document.getElementById("directions"));

// create a marker for each address
createMarker({
      name: 'Western Avenue Office (Corporat',
      html: '<p><b>Western Avenue Office (Corporat</b><br>4259 N. Wes
      lon: -87.688417,lat:41.959008,
      city: 'Chicago'
});
```

Dissecting map.js

```
// create the map
map = new GMap2(document.getElementById("map"));
...
// === create a GDirections Object ===
gdir=new GDirections(map, document.getElementById("directions"));

// create a marker for each address
createMarker({
      name: 'Western Avenue Office (Corporat',
      html: '<p><b>Western Avenue Office (Corporat</b><br>4259 N. Wes
      lon: -87.688417,lat:41.959008,
      city: 'Chicago'
});
```

Dissecting map.js

```
// create the map
map = new GMap2(document.getElementById("map"));
...
// === create a GDirections Object ===
gdir=new GDirections(map, document.getElementById("directions"));

// create a marker for each address
createMarker({
        name: 'Western Avenue Office (Corporat',
        html: '<p><b>Western Avenue Office (Corporat</b><br>4259 N. Wes
        lon: -87.688417,lat:41.959008,
        city: 'Chicago'
});
```

Dissecting map.js

Dissecting map.js

```
function geoSearch() {
  var zipcode = $F('zipcode');
  var miles = parseInt($F('miles'));
  side_bar_html = "";
  if (zipcode.length > 0) {
    // geocode zipcode
    var geocoder = new GClientGeocoder();
    geocoder.getLatLng(
      zipcode,
      function(geoPoint) {
        if (geoPoint) {
          point = geoPoint;
        } else {
          point = gmarkers[0].getPoint();
        }
        map.clearOverlays();
        for (var j = 0; j < gmarkers.length; j++) {
          var distKm=gmarkers[j].getPoint().distanceFrom(point)/1000;
          var distMiles = distKm * 0.621371;
          if (distMiles <= miles) {
            map.addOverlay(gmarkers[j]);
            side_bar_html += '<a href="javascript:myclick(' + j + ')">' + agents[j].name + '<\
          }
        }
        switch (miles) {
        case 1: newZoom = 14; break;
        case 5: newZoom = 12; break;
        case 10: newZoom = 10; break;
```

Dissecting map.js

```
function geoSearch() {
  var zipcode = $F('zipcode');
  var miles = parseInt($F('miles'));
  side_bar_html = "";
  if (zipcode.length > 0) {
    // geocode zipcode
    var geocoder = new GClientGeocoder();
    geocoder.getLatLng(
      zipcode,
      function(geoPoint) {
        if (geoPoint) {
          point = geoPoint;
        } else {
          point = gmarkers[0].getPoint();
        }
        map.clearOverlays();
        for (var j = 0; j < gmarkers.length; j++) {
          var distKm=gmarkers[j].getPoint().distanceFrom(point)/1000;
          var distMiles = distKm * 0.621371;
          if (distMiles <= miles) {
            map.addOverlay(gmarkers[j]);
            side_bar_html += '<a href="javascript:myclick(' + j + ')">' + agents[j].name + '<\
          }
        }
        switch (miles) {
        case 1: newZoom = 14; break;
        case 5: newZoom = 12; break;
        case 10: newZoom = 10; break;
```

Dissecting map.js

```
function geoSearch() {
  var zipcode = $F('zipcode');
  var miles = parseInt($F('miles'));
  side_bar_html = "";
  if (zipcode.length > 0) {
    // geocode zipcode
    var geocoder = new GClientGeocoder();
    geocoder.getLatLng(
      zipcode,
      function(geoPoint) {
        if (geoPoint) {
          point = geoPoint;
        } else {
          point = gmarkers[0].getPoint();
        }
        map.clearOverlays();
        for (var j = 0; j < gmarkers.length; j++) {
          var distKm=gmarkers[j].getPoint().distanceFrom(point)/1000;
          var distMiles = distKm * 0.621371;
          if (distMiles <= miles) {
            map.addOverlay(gmarkers[j]);
            side_bar_html += '<a href="javascript:myclick(' + j + ')">' + agents[j].name + '<\
          }
        }
        switch (miles) {
        case 1: newZoom = 14; break;
        case 5: newZoom = 12; break;
        case 10: newZoom = 10; break;
```

Dissecting map.js

```
function geoSearch() {
  var zipcode = $F('zipcode');
  var miles = parseInt($F('miles'));
  side_bar_html = "";
  if (zipcode.length > 0) {
    // geocode zipcode
    var geocoder = new GClientGeocoder();
    geocoder.getLatLng(
      zipcode,
      function(geoPoint) {
        if (geoPoint) {
          point = geoPoint;
        } else {
          point = gmarkers[0].getPoint();
        }
        map.clearOverlays();
        for (var j = 0; j < gmarkers.length; j++) {
          var distKm=gmarkers[j].getPoint().distanceFrom(point)/1000;
          var distMiles = distKm * 0.621371;
          if (distMiles <= miles) {
            map.addOverlay(gmarkers[j]);
            side_bar_html += '<a href="javascript:myclick(' + j + ')">' + agents[j].name + '<\
          }
        }
        switch (miles) {
        case 1: newZoom = 14; break;
        case 5: newZoom = 12; break;
        case 10: newZoom = 10; break;
```

Dissecting map.js

```
function geoSearch() {
  var zipcode = $F('zipcode');
  var miles = parseInt($F('miles'));
  side_bar_html = "";
  if (zipcode.length > 0) {
    // geocode zipcode
    var geocoder = new GClientGeocoder();
    geocoder.getLatLng(
      zipcode,
      function(geoPoint) {
        if (geoPoint) {
          point = geoPoint;
        } else {
          point = gmarkers[0].getPoint();
        }
        map.clearOverlays();
        for (var j = 0; j < gmarkers.length; j++) {
          var distKm=gmarkers[j].getPoint().distanceFrom(point)/1000;
          var distMiles = distKm * 0.621371;
          if (distMiles <= miles) {
            map.addOverlay(gmarkers[j]);
            side_bar_html += '<a href="javascript:myclick(' + j + ')">' + agents[j].name + '<\
          }
        }
        switch (miles) {
        case 1: newZoom = 14; break;
        case 5: newZoom = 12; break;
        case 10: newZoom = 10; break;
```

Dissecting map.js

Dissecting map.js

- So far, we've created a map object

Dissecting map.js

- So far, we've created a map object
- Added a search function

Dissecting map.js

- So far, we've created a map object
- Added a search function
- Placed markers on the page within our search radius

Dissecting map.js

- So far, we've created a map object
- Added a search function
- Placed markers on the page within our search radius
- Now, let's add directions

Dissecting map.js

```
    GEvent.addListener(marker, "click", function() {
      marker.openInfoWindowHtml(html);
    });
    // save the info we need to use later for the side_bar
    gmarkers.push(marker);
agents.push(agent);
    htmls[i] = html;
    return marker;
  }

  // ===== request the directions =====
function getDirections(i) {
  // ==== Set up the walk and avoid highways options ====
  var opts = {};
  if (document.getElementById("walk").checked) {
     opts.travelMode = G_TRAVEL_MODE_WALKING;
  }
  if (document.getElementById("highways").checked) {
     opts.avoidHighways = true;
  }
  // ==== set the start and end locations ====
  var saddr = document.getElementById("saddr").value
  var daddr = document.getElementById("daddr").value
  gdir.load("from: "+saddr+" to: "+daddr, opts);
}
```

Dissecting map.js

```
    GEvent.addListener(marker, "click", function() {
      marker.openInfoWindowHtml(html);
    });
    // save the info we need to use later for the side_bar
    gmarkers.push(marker);
agents.push(agent);
    htmls[i] = html;
    return marker;
  }

  // ===== request the directions =====
  function getDirections(i) {
    // ==== Set up the walk and avoid highways options ====
    var opts = {};
    if (document.getElementById("walk").checked) {
       opts.travelMode = G_TRAVEL_MODE_WALKING;
    }
    if (document.getElementById("highways").checked) {
       opts.avoidHighways = true;
    }
    // ==== set the start and end locations ====
    var saddr = document.getElementById("saddr").value
    var daddr = document.getElementById("daddr").value
    gdir.load("from: "+saddr+" to: "+daddr, opts);
  }
```

Dissecting map.js

```
    GEvent.addListener(marker, "click", function() {
      marker.openInfoWindowHtml(html);
    });
    // save the info we need to use later for the side_bar
    gmarkers.push(marker);
agents.push(agent);
    htmls[i] = html;
    return marker;
  }

  // ===== request the directions =====
  function getDirections(i) {
    // ==== Set up the walk and avoid highways options ====
    var opts = {};
    if (document.getElementById("walk").checked) {
       opts.travelMode = G_TRAVEL_MODE_WALKING;
    }
    if (document.getElementById("highways").checked) {
       opts.avoidHighways = true;
    }
    // ==== set the start and end locations ====
    var saddr = document.getElementById("saddr").value
    var daddr = document.getElementById("daddr").value
    gdir.load("from: "+saddr+" to: "+daddr, opts);
  }
```

Dissecting map.js

```
    GEvent.addListener(marker, "click", function() {
      marker.openInfoWindowHtml(html);
    });
    // save the info we need to use later for the side_bar
    gmarkers.push(marker);
agents.push(agent);
    htmls[i] = html;
    return marker;
  }

  // ===== request the directions =====
  function getDirections(i) {
    // ==== Set up the walk and avoid highways options ====
    var opts = {};
    if (document.getElementById("walk").checked) {
       opts.travelMode = G_TRAVEL_MODE_WALKING;
    }
    if (document.getElementById("highways").checked) {
       opts.avoidHighways = true;
    }
    // ==== set the start and end locations ====
    var saddr = document.getElementById("saddr").value
    var daddr = document.getElementById("daddr").value
    gdir.load("from: "+saddr+" to: "+daddr, opts);
  }
```

Dissecting map.js

```
    GEvent.addListener(marker, "click", function() {
      marker.openInfoWindowHtml(html);
    });
    // save the info we need to use later for the side_bar
    gmarkers.push(marker);
agents.push(agent);
    htmls[i] = html;
    return marker;
  }

  // ===== request the directions =====
  function getDirections(i) {
    // ==== Set up the walk and avoid highways options ====
    var opts = {};
    if (document.getElementById("walk").checked) {
       opts.travelMode = G_TRAVEL_MODE_WALKING;
    }
    if (document.getElementById("highways").checked) {
       opts.avoidHighways = true;
    }
    // ==== set the start and end locations ====
    var saddr = document.getElementById("saddr").value
    var daddr = document.getElementById("daddr").value
    gdir.load("from: "+saddr+" to: "+daddr, opts);
  }
```

Code Walkthrough

Credits/Resources

- Google's Map API
  http://code.google.com/apis/maps/
- Google Maps Tutorial
  http://econym.org.uk/gmap/